# Embedded Systems International

# Lab 6 Prelab

**Name**: Jason Di Giovanni

**Lab Partner Name (if you worked together and are submitting the same document or mostly the same answers):**

**Lab Section**:

**Submit your prelab document as a PDF file in Canvas under the corresponding prelab assignment. Every student submits their own prelab. Lab partners are allowed to work on the prelab together and submit the same document (if there is actual collaboration on the document). For full credit, the prelab must be submitted prior to the start of lab. Text responses should be typed or printed neatly. You can draw a sketch by hand, or you can use a drawing tool. Try to have started a rough draft of the prelab when you come to class on Tuesday.**

Questions 1-7 were included in the Lab 5 Prelab. You selected 4 questions to be graded. The other 3 questions will be graded for this prelab. Circle the questions selected for Prelab 5 and Prelab 6. If none are circled, questions 5-7 will be graded. The same questions cannot be circled for both prelabs 5 and 6. Help your TA by circling the questions properly.

**4 questions selected and graded for Prelab 5**: 1 2 3 4 5 6 7

**3 questions selected and graded for Prelab 6**: 1 2 3 4 5 6 7

**In addition, answer the remaining questions, 8 - 10, in this prelab.**

1. Port pins for UART1 transmit (TX) and receive (RX)

UART1 is an alternate function, or peripheral, that uses GPIO Port B pins.

a) Which Port B pin is used for the UART1 transmit (TX) signal?

b) Which Port B pin is used for the UART1 receive (RX) signal?

2. Use the GPIO port worksheet below to indicate which Port B pin is used for UART1 TX and which Port B pin is used for UART1 RX: write TX and RX in the DATA row of the worksheet under the corresponding pin numbers. What values must be initialized in the other GPIO port registers to configure the UART1 alternate function for the port. Put an "x" in cells that you are not using and should preserve their values.

GPIO Port Registers

| Bit >>><br>vvv Register | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Description |
|---|---|---|---|---|---|---|---|---|---|
| DATA<br>(or PORT pins for alternate functions) | x | x | x | x | x | x | 1 | 1 | 0, 1: bit values<br>(or alternate functions) |
| DEN | x | x | x | x | x | x | 1 | 1 | 1: enable digital |
| AFSEL | x | x | x | x | x | x | 1 | 1 | 1: enable alternate function |
| PCTL | x | x | x | x | x | x | 0x1 | 0x1 | Each column in this row is a 4-bit field in the 32-bit PCTL register |

Refer as needed to the examples on pages 19-28 of the mtg10-notes-concepts-GPIOaltfunc-UART.pdf file and Bai book Table 8.1 (in the notes and slides) (see also Tiva datasheet section 23.4 GPIO Pins and Alternate Functions, Table 23-5, for PMCx Bit Field Encoding).

3. Refer to the GPIOAFSEL register description in the Tiva datasheet (Register 10). Refer to the macro definition for the GPIO Port B AFSEL register in the header file, tm4c123gh6pm.h.

   a. Write the full #define macro definition for the PORTB_AFSEL register as given in the header file.
   b. What is the 32-bit memory address of this AFSEL register (in hex)?
   c. Explain how to determine the register address, i.e., your answer to part b (the memory address for the Port B AFSEL register), from the Tiva datasheet register description (pages 671-672).

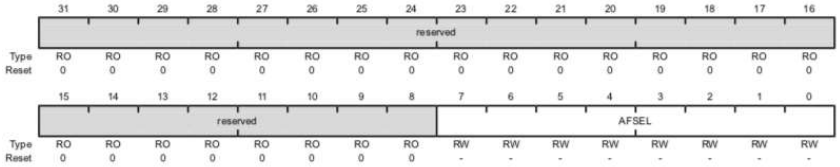
      Hint: Pay attention to the following information from the datasheet pages. Recall that we are using the APB bus on the microcontroller.

**Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420**

The **GPIOAFSEL** register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The **GPIO Port Control (GPIOPCTL)** register is used to select one of the possible functions. Table 23-5 on page 1351 details which functions are muxed on each GPIO pin. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x420
Type RW, reset -



4. For a baud rate of 115,200, what values should be loaded into the IBRD and FBRD registers?

UART Integer Baud Rate Divisor (UARTIBRD), e.g., UART1_IBRD_R
UART Fractional Baud Rate Divisor (UARTFBRD), e.g., UART1_FBRD_R

Note: A 16 MHz system clock for the TM4C123 microcontroller is used in lab. Also ClkDiv = 16.

Refer to any of the following resources about setting the baud rate for a UART on the TM4C123:

- In the Bai book, see sections 8.5.3.7 and 8.5.5.2 (also in mtg10-slides-UART.pdf, page 29).
- In VY ES book, see section 11.2.2 and information about IBRD and FBRD (remember, we are using the 16 MHz system clock).
- In the Tiva datasheet, see section 14.3.2.

5. Consider the UART Line Control register (UARTLCRH).

   a. Look up this register in the Tiva datasheet (p. 916, try to find it without using the page number). Read the descriptions for the bits. Identify which bits correspond to the following serial communication parameters:

8 data bits, 1 stop bit, no parity, FIFOs disabled, no interrupts

For these parameter values, fill in the register bit values in this blank register (assume default/reset values if not specified):

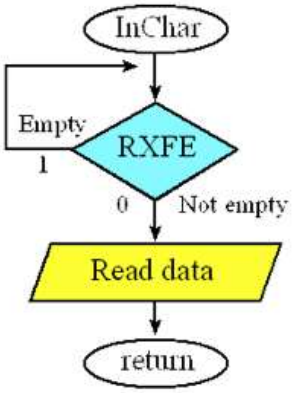
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Value | | | | | | | | |

Write a line of code to initialize the UART Line Control register **:**

```
UART1_LCRH_R =
```

   b. What is the 32-bit memory address of UART1 LCRH register (in hex)?

6. Consider the flowchart from Figure 11.8 in section 11.2.3 in Chapter 11 of Valvano and Yerraballi ES online book, http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C11_SerialInterface.htm . This shows a busy-wait or polling approach to receiving data by a UART.




The RXFE flag is a bit in one of the UART registers. What is the name of the register? What is the bit number for the RXFE flag?

UART Flag Register bit 4

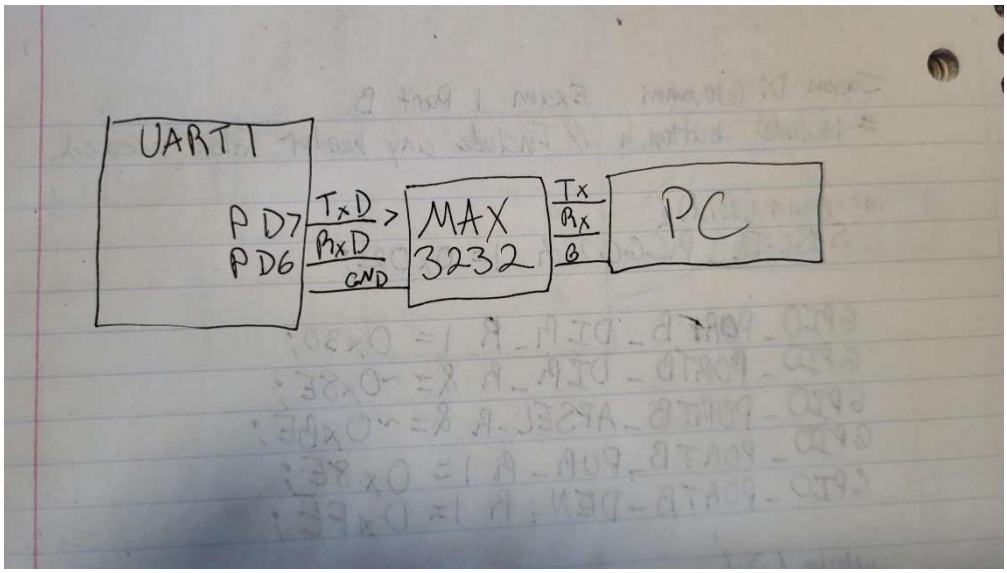
7. System information - sketch

Refer to Video 12.2a (UART Background and Launchpad Support) in Chapter 11 of Valvano and Yerraballi ES online book, http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C11_SerialInterface.htm (section 11.2 UART).

Video 12.2a: https://www.youtube.com/watch?v=wdOqQCNWC3c&feature=youtu.be

The authors sketch three scenarios for using a UART on the TM4C123 LaunchPad board. Note that we have a version of the third scenario that uses a MAX3232 driver chip (you can see this chip in the Cybot baseboard schematic, photo, or actual board near the DB9 serial socket).

Sketch your own version of the third scenario, similar to that shown in the video. Your sketch should specify UART1 and the actual GPIO port pins used to communicate with the PC in the lab.



**Answer the remaining prelab questions, 8 - 10.**

8. I/O synchronization behavior

Read section 11.1, I/O Synchronization, in the VYES book, stopping before Interactive Tool 11.3 http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C11_SerialInterface.htm (note that the interactive tools in this chapter may only work in the Firefox browser, and if so, ignore them). Read about the synchronization mechanisms shown in figures 11.1 and 11.2. See also Figure 12.2 from the VYES book, which shows the third (interrupt) mechanism as well as how a main program executes other tasks.

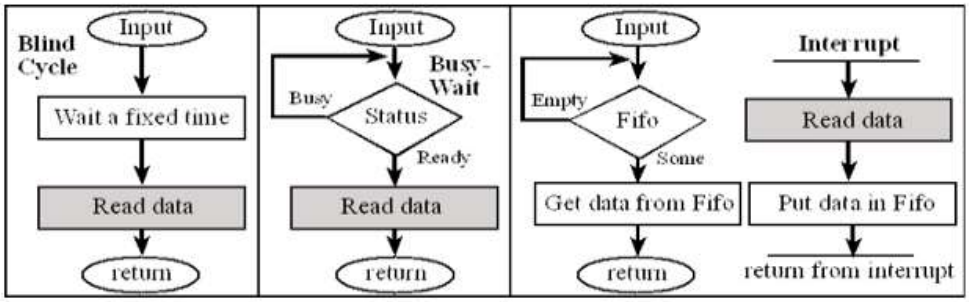



*Figure 11.1. Synchronization Mechanisms*

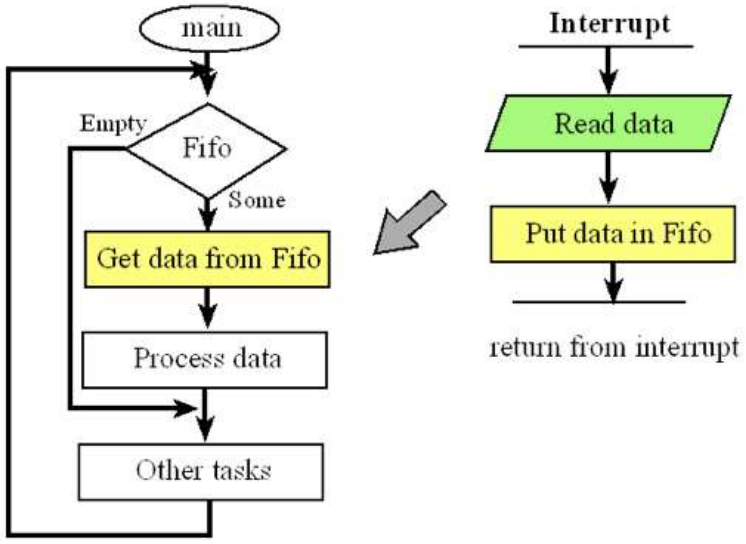



*Figure 12.2. For an input device we can use a FIFO to pass data from the ISR to the main program.*

Instead of polling (i.e., waiting in a loop) to receive data, an interrupt approach can be used. This lets the main program do other tasks. A separate interrupt handler function (interrupt service routine, ISR) is executed whenever a character is received by a UART. The main program doesn't have to busy-wait and instead just gets data when it needs it.

For example, think about an analogy of a classroom: suppose the instructor is like a processor executing a main program of teaching, including receiving student questions. Briefly explain the difference between polling and interrupts with this or another similar analogy. For example, explain how the instructor receives questions in relation to doing other tasks when a) polling, or b) using interrupts.

When using polling, the instructor (processor) continuously scans the room (checks a register or flag) to see if any students (UART) have raised their hands (data ready). The instructor stops teaching (main program execution) momentarily to scan the room (busy-waiting), and only when a student raises their hand (data is ready), the instructor pauses the lecture (main program execution) to address the question (process the received data). During this time, the instructor cannot continue with the lecture until a question is asked (data is received), potentially leading to inefficiency as the instructor's teaching time is interrupted by frequent checks for questions.

Using interrupts is akin to having a class monitor (interrupt handler) who alerts the instructor (processor) whenever a student (UART) raises their hand (data is received). The instructor can continue teaching (main program execution) without interruption, and whenever a question is asked (data is received), the class monitor (interrupt handler) immediately notifies the instructor (processor), who then pauses the lecture briefly to address the question (process the received data). The key difference is that the instructor (processor) does not actively check for questions (data) but rather continues with other tasks until notified by the class monitor (interrupt handler) that there is something to address, leading to more efficient use of time (processor resources) as the instructor can focus on teaching (main program execution) uninterrupted until needed.

9. I/O synchronization - UART registers

Consider some of the UART registers used when programming the mechanisms shown in question 8.

UART Data (UARTDR): UARTx_DR_R
UART Flag (UARTFR): UARTx_FR_R
UART Interrupt Mask (UARTIM): UARTx_IM_R
UART Raw Interrupt Status (UARTRIS): UARTx_RIS_R
UART Masked Interrupt Status (UARTMIS): UARTx_MIS_R
UART Interrupt Clear (UARTICR): UARTx_ICR_R

Refer to VYES Figure 11.1. Assume we are using UART1 as in Lab 6. Identify one register from the list above for each question below. A register will be used for only one answer.

a. Which register would be read in the "Read data" box in Figure 11.1?

UARTx_DR_R

b. Which register would be tested in the “Status” decision box in the figure?

UARTx_FR_R

c. Which register indicates that there has been a trigger for an interrupt, such as receiving a character into the UART (one of the conditions necessary to get to the “Interrupt” in the figure)?

UARTx_RIS_R

d. Which register is set during initialization to allow (or enable) an interrupt signal from the UART (one of the conditions necessary to get to the “Interrupt” in the figure)?

UARTx_IM_R

10. UART Interrupt Mask Register

Refer to the UARTIM register description in the Tiva datasheet. Refer to the macro definition for the UART1 Interrupt Mask register in the header file, tm4c123gh6pm.h.

d. Write the full #define macro definition as given in the header file.

#define UART1_IM_R (*((volatile unsigned long *)0x4000D038))

e. Write the line of code for the IM register to enable RX interrupts from the UART1 device.

```
UART1_IM_R |= 0x10;
```